
Bayesian Neural Networks for Genetic Association Studies of Complex Disease

Andrew L. Beam*
 Bioinformatics Research Center
 North Carolina State University
 Raleigh, NC
 albeam@ncsu.edu

Alison Motsinger-Reif
 Bioinformatics Research Center
 Department of Statistics
 North Carolina State University
 Raleigh, NC
 alison.motsinger@ncsu.edu

Jon Doyle
 Department of Computer Science
 North Carolina State University
 Raleigh, NC
 Jon.Doyle@ncsu.edu

Abstract

Background

Discovering causal genetic variants from large genetic association studies poses many difficult challenges. Assessing which genetic markers are involved in determining trait status is a computationally demanding task, especially in the presence of gene-gene interactions.

Results

A non-parametric Bayesian approach in the form of a Bayesian neural network is proposed for use in analyzing genetic association studies. Demonstrations on synthetic and real data reveal they are able to efficiently and accurately determine which variants are involved in determining case-control status. Using graphics processing units (GPUs) the time needed to build these models is decreased by several orders of magnitude. In comparison with commonly used approaches for detecting interactions, Bayesian neural networks perform very well across a broad spectrum of possible genetic relationships.

Conclusions

The proposed framework is shown to be powerful at detecting causal SNPs while having the computational efficiency needed to handle large datasets.

1 Background

The ability to rapidly collect and genotype large numbers of genetic variants has outpaced the ability to interpret such data, leaving the genetic etiology for many diseases incomplete. The presence of gene-gene interactions, or epistasis, is believed to be a critical piece of this missing heritability [Manolio et al., 2009]. This has in turn spurred development on advanced computational approaches to account for these interactions, with varying degrees of success [Motsinger-Reif et al., 2008b, Motsinger-Reif et al., 2008a, Koo et al., 2013]. The main computational challenge comes from the

*Corresponding author.

vast number of markers that are present in a typical association study. This problem is exacerbated when interactions between two or more markers must be considered. For example, given an experiment that genotypes 1,000 markers, examining all possible interactions between two of the markers involves consideration of nearly half a million combinations. This situation becomes exponentially worse as higher order interactions are considered. Modern genome-wide association studies (GWASs) routinely consider 1-2 million single nucleotide polymorphisms (SNPs), which would require examining half a trillion potential interactions. As whole genome sequencing (WGS) methods become commonplace, methods that cannot cope with large data sets will be of little utility. Data on this scale will require approaches that can find interactions without having to enumerate all possible combinations. As genotypic technology advances, datasets now routinely include millions of SNPs.

Several distinct types of methods have emerged that attempt to address this challenge. Perhaps one of the most popular approaches from the last decade has been Multifactor Dimensionality Reduction (MDR) [Moore et al., 2006, Hahn et al., 2003], and extensions of the method. MDR is a combinatorial search that considers all possible interactions of a given order and selects the best model via cross validation. Because MDR is an exhaustive search, it suffers from the previously discussed scalability issue, though recent work using graphics processing units has attempted to lessen this deficit [Greene et al., 2010]. MDR is reliant upon a permutation testing strategy to assess statistical significance for each marker, so the computational burden becomes prohibitive for large datasets. Permutation testing computes a p-value for a statistic of interest (such as an accuracy measure from MDR) by randomly permuting the class labels and calculating the statistic on the permuted dataset. This procedure is repeated many times to compute a null distribution for the statistic of interest. The relative percentage of instances in the permuted null distribution that are less than or equal to the actual statistic from the unpermuted data is taken as the desired one-sided p-value. Unfortunately, this can be extremely expensive for large datasets when many hypotheses are simultaneously tested, leading to a large multiple testing scenario. To get the required resolution for a Bonferroni corrected p-value of 0.05 when considering a set of 1,000 SNPs, one must perform 20,000 permutations. This makes permutation testings infeasible for even moderately sized datasets.

Another popular approach is Bayesian Epistasis Association Mapping (BEAM) [Zhang and Liu, 2007]. BEAM partitions markers into groups representing individual (i.e. marginal) genetic effects, interactions, and a third group representing background markers that are uninvolved with the trait. BEAM employs a stochastic Markov Chain Monte Carlo (MCMC) search technique to probabilistically assign markers to each group and uses a novel B-statistic based on the MCMC simulation to assign statistical significance to each marker. This allows BEAM to assign statistical significance without the need to perform a costly permutation test. This method has been demonstrated successfully on data sets with half a million markers. However, the recommended amount of MCMC iterations needed is quadratic in the number of SNPs considered [Zhang and Liu, 2007], possibly limiting its effectiveness for larger datasets.

Many popular machine learning algorithms have also been adopted for use in analyzing association studies. Notable examples are decision trees (both bagged, i.e. random forests, [Lunetta et al., 2004, Diaz-Uriarte and de Andres, 2006] and boosted [Li et al., 2011] support vector machines (SVM) [Guyon et al., 2002], Bayesian networks [Jiang et al., 2011], and neural networks [Motsinger-Reif et al., 2008b]. In particular, tree-based methods such as random forests and boosted decision trees have been found to perform well in several previous association studies [Lunetta et al., 2004, Li et al., 2011, Diaz-Uriarte and de Andres, 2006]. Machine learning approaches are appealing because they assume very little *a priori* about the relationship between genotype and phenotype, with most methods being flexible enough to model complex relationships accurately. However, this generality is something of a double-edged sword as many machine learning algorithms function as black boxes, providing investigators with little information on which variables may be most important. Typically it is the goal of an association study to determine which variables are most important, so a black box may be of little use. Some approaches have easy adaptations that allow them to provide such measures. Both types of tree based methods (bagged and boosted) can provide measures of relative variable importance [Breiman, 2001, Friedman, 2001], but these indicators lack measures of uncertainty, so they are unable to determine how likely a variable's importance measure is to occur by chance without resorting to permutation testing.

In this study, we propose the use of Bayesian neural networks (BNNs) for association studies to directly address some the issues with current epistasis modeling. While BNNs have been previ-

ously developed and applied for other tasks [Lisboa et al., 2003, Baesens et al., 2002, Neal, 1995, Neal, 1992] they have yet to see significant usage in bioinformatics and computational biology. Like most complex Bayesian models, BNNs require stochastic sampling techniques that draw samples from the posterior distribution, because direct or deterministic calculation of the posterior distribution is often intractable. These posterior samples are then used to make inferences about the parameters of the model or used to make predictions for new data. Standard MCMC methods that employ a random walk such as the Metropolis-Hastings (RW-MH) algorithm [Metropolis et al., 2004, Hastings, 1970] (which is the algorithm that forms the core of BEAM [Zhang and Liu, 2007]) explores the posterior distribution very slowly when the number of predictors is large. If d is the number of parameters in a model, the number of iterations needed to obtain a nearly independent sample is $O(d^2)$ [Neal, 2011b] for RW-MH. This makes the RW-MH algorithm unsuitable for neural network models in high-dimensions, so the Hamiltonian Monte Carlo (HMC) algorithm is instead used to generate samples from the posterior. HMC has more favorable scaling properties, as the number of iterations needed is only $O(d^{5/4})$ [Neal, 2011b]. HMC achieves this favorable scaling by using information about the gradient of the log-posterior distribution to guide the simulation to regions of high posterior probability. Readers familiar with standard neural network models will notice an inherent similarity between Bayesian neural networks sampled using HMC and traditional feed-forward neural networks that are trained using the well known back-propagation algorithm [Rumelhart et al., 1988], as both take steps in the steepest direction using gradient based information. Though HMC will in general explore the posterior distribution in a more efficient manner than RW-MH, the evaluation of the gradient can very expensive for large data sets. Recent work has shown that this drawback can be lessened through the use of parallel computing techniques [Beam et al., 2014].

The BNN framework outlined here has several features designed to address many of the challenges inherent in analyzing large datasets from genetic association studies. These advantages are outlined below.

- Quantification of variable influence with uncertainty measures. This allows variable influence to be assessed relative to a null or background model using a novel Bayesian testing framework. This avoids reliance on a permutation testing strategy.
- Automatic modeling of arbitrarily complex genetic relationships. Interactions are accounted for without having to examine all possible combinations. This is achieved from the underlying neural network model.
- An efficient sampling algorithm. HMC scales much better than other MCMC methods, such as the RW-MH algorithm, in high-dimensions.
- Computational expediency through the use of GPUs. The time needed to build the model is greatly reduced using the massive parallel processing offered by GPUs.

We offer evidence for these claims using several simulated scenarios and a demonstration on a real GWAS dataset. In addition, we compare the proposed approach to several popular methods so that relative performance can be assessed.

2 Methods

Neural networks are a set of popular methods in machine learning that have enjoyed a flurry of renewed activity spurred on by advances in training so-called deep networks [Hinton et al., 2012, Bengio, 2009]. The term neural network can refer to a very large class of modeling techniques, but to be clear we use the term here to refer to multilayer feed-forward perceptions (MLPs). In the most basic sense, neural nets represent a class of non-parametric methods for regression and classification. They are non-parametric in the sense that they are capable of modeling any smooth function on a compact domain to an arbitrary degree of precision without the need to specify the exact relationship between input and output. This is often succinctly stated as neural nets are *universal function approximators* [Hornik et al., 1989]. This property makes them appealing for many tasks, including modeling the relationship between genotype and phenotype, because a sufficiently complex network will be capable of automatically representing the underlying function.

Some draw backs of classical neural nets are natural consequences of their strengths. Due to their flexibility, neural nets are highly prone to over-fitting to data used to train them. Over-fitting occurs

when the network starts to represent the training data exactly, including noise that may be present, which reduces its ability to generalize to new data. Many methods exist to address this issue, but popular methods such as weight decay are well known to be approximations to a fully Bayesian procedure [Neal, 1995, Williams, 1995]. Another issue with standard neural nets is they are often regarded as black boxes in that they do not provide much information beyond a predicted value for each input. Little intuition or knowledge can be gleaned as to which inputs are most important in determining the response, so nothing coherent can be said as to what drives the networks predictions. Discussions of the advantages and disadvantages of neural nets for gene mapping have been reviewed in [Motsinger-Reif and Ritchie, 2008]. First we describe the base neural network model, and then describe how this can be incorporated into a Bayesian framework.

The network is defined in terms of a directed, acyclic graph (DAG) where inputs are feed into a layer of hidden units. The output of the hidden units are then fed in turn to the output layer which transforms a linear combination of the hidden unit outputs into a probability of class membership. Specifically consider a network with p input variables, h hidden units, and 2 output units to be used to predict whether an observation belongs to class 1 or class 2. Let $x_i = \langle x_{i1}, \dots, x_{ip} \rangle^T$ be the input vector of p variables and $y_i = \langle y_{i1}, y_{i2} \rangle$ be the response vector, where $y_{i1} = 1$ if observation i belongs to class 1 and 0 if not, with y_{i2} is defined in the same way for class 2. Hidden unit k first takes a linear combination of each input vector followed by a nonlinear transformation, using the following form:

$$h_k(x_i) = \phi \left(b_k + \sum_{j=1}^p w_{kj} * x_{ij} \right) \quad (1)$$

where $\phi(\cdot)$ is a nonlinear function. For the purposes of this study, consider the logistic transformation, given as:

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Several other activation functions such as the hyperbolic tangent, linear rectifier, and the radial basis/Gaussian functions are often used in practice. Each output unit takes a linear combination of each h_k followed by another nonlinear transformation. Let $f_1(x_i)$ be the output unit that is associated with class 1:

$$f_1(x_i) = \psi \left(B_1 + \sum_{k=1}^h W_{k1} * h_k(x_i) \right) \quad (2)$$

Note we have used upper case letters to denote parameters in the output layer and lowercase letters to indicate parameters belonging to the hidden layer. The $\psi(\cdot)$ function is the softmax transformation of element z_1 from the vector $z = \langle z_1, \dots, z_n \rangle$:

$$\psi(z_1) = \frac{\exp(z_1)}{\sum_{i=1}^n \exp(z_i)}$$

In this representation, $f_1(x_i)$ represents the estimated conditional probability that y_i belongs to class 1, given the input vector x_i . A similar definition is made for output unit 2, $f_2(x_i)$. Note that for the case of only 2 classes, $f_2(x_i) = 1 - f_1(x_i)$ because the softmax transformation forces the outputs to sum to 1.

Having described the formulation for standard neural networks we next describe how this can be extended using the Bayesian formulation. Bayesian methods define a probability distribution over possible parameter values, and thus over possible neural networks. To simplify notation, let $\theta = \{B, W, \beta, w\}$ represent all of the network weights and biases shown in equations 1, 2. The posterior distribution for θ given the data x_i and y_i , is given according to Bayes rule:

$$p(\theta|x_i, y_i) = \frac{L(\theta|x_i, y_i) \cdot \pi(\theta)}{m(x)} \quad (3)$$

where $m(x) = \int L(\theta|x_i, y_i) \cdot \pi(\theta) d\theta$ is the marginal density of the data. $L(\theta|x_i, y_i)$ is the *likelihood* of θ given then data and $\pi(\theta)$ is the prior distribution for the network parameters. However, in practice we only need to be able to evaluate the numerator of (3) up to a constant because we will be relying on MCMC sampling techniques that draw from the correct posterior without having to evaluate $m(x)$, which may be intractable in high dimensions.

Often it is better to work with the log-likelihood $l(\theta|x_i, y_i) = \log(L(\theta|x_i, y_i))$, because the raw likelihood can suffer from numerical overflow or underflow for large problems. In this study we assume the log-likelihood for a neural network with 2 output units is binomial:

$$l(\theta|x_i, y_i) = y_{i1} \cdot \log(f_1(x_i)) + y_{i2} \cdot \log(1 - f_1(x_i)) \quad (4)$$

Next every parameter in the model must be given a prior distribution. The prior distribution codifies beliefs about the values each parameter is likely to take before seeing the data. This type of formulation is extremely useful in high-dimensional settings such as genomics, because it enables statements such as ‘most of the variables are likely to be unrelated to this response’ to be quantified and incorporated into the prior. In this study, we adopt a specific prior structure known as the *Automatic Relevance Determination* (ARD) prior. This prior was originally introduced in some of the foundational work on Bayesian neural nets [Neal, 1995, Neal, 1998] and later used in SVMs for cancer classification [Li et al., 2002].

The ARD prior groups network weights in the hidden layer together in a meaningful and interpretable way. All of the weights in the hidden layer that are associated with the same input variable are considered part of the same group. Each weight in a group is given a normal prior distribution with mean zero and a common variance parameter. This shared group-level variance parameter controls how large the weights in a group are allowed to become and performs shrinkage by pooling information from several hidden units, which helps to prevent overfitting [Neal, 1998]. Each of the group-level variance parameters is itself given a prior distribution, typically an Inverse-Gamma distribution with some shape parameter α_0 and some scale parameter β_0 . These parameters often referred to as hyper parameters, and can themselves be subject to another level of prior distributions, but for the purposes of this study, we will leave them fixed as user specified values. Specifically, for a network with h hidden units, the weights in the hidden layer for input variable j will have the following prior specification:

$$w_{j1}, \dots, w_{jp} \sim N(0, \sigma_j^2) \\ \sigma_j^2 \sim IG(\alpha_0, \beta_0)$$

This structure allows the network to automatically determine which of the inputs are most relevant. If variable j is not very useful in determining whether an observation is a case or a control, then the posterior distribution for σ_j^2 will be concentrated around small values. Likewise, if variable j is useful in determining the response status, most of the posterior mass for σ_j^2 will be centered on larger values.

2.1 Hamiltonian Monte Carlo (HMC) for Neural Networks

Here we briefly give an overview of Hamiltonian Monte Carlo (HMC) for neural networks, but please see [Neal, 1995, Neal, 1992, Neal, 2011a] for a thorough treatment. HMC is one of many Markov Chain Monte Carlo (MCMC) methods used to draw samples from probability distributions that may not have analytic closed forms. HMC is well suited for high-dimensional models, such as neural nets, because it uses information about the gradient of the log-posterior to guide the sampler to regions of high posterior probability. For neural networks, we adopt the two-phase sampling scheme of Neal used in [Neal, 1995]. In the first phase, we update the values of the variance parameters using a Gibbs update, conditional on the current values of the networks weights. In the next phase, we leave the variance parameters fixed and update the network weights using HMC. We repeat this procedure of Gibbs-coupled HMC updates until we have acquired the desired number of posterior samples.

The higher level variance parameters, including those in the ARD prior, have simple closed forms because the Inverse-Gamma distribution is a conditionally conjugate prior distribution for the variance parameter of a Normal distribution. To obtain a new value for a variance parameter, conditional on

the values of the weights a parameter controls, one makes a draw from the following Inverse-Gamma distribution:

$$\sigma_{new}^2 \sim IG\left(\alpha_0 + \frac{n_w}{2}, \beta_0 + \sum_{i=1}^{n_w} \frac{w_i^2}{2}\right) \quad (5)$$

where each w_i is a weight controlled by this variance parameter, n_w is the number of weights in the group, and α_0, β_0 are the shape and scale parameters respectively of the prior distribution.

HMC then proceeds by performing L number of leap-frog updates for the weights, given the values of the variance parameters. The algorithm introduces a fictitious momentum variable for every parameter in the network that will be updated by simulating Hamiltonian dynamics on the surface of the log-posterior. Since HMC was originally created in statistical physics it is often presented in terms of energy potentials which is equivalent to an exponentiation of the negative log-posterior, but we will describe the algorithm directly in terms of the log-posterior, which is more natural for our purposes. The full algorithm for sampling the posterior of all parameters (network weights and variance parameters) is shown below.

HMC Algorithm for Neural Networks

Input: **X**: matrix of predictors **Y**: matrix of class membership

P: log-posterior density to be sampled

$\{\epsilon, L, N_s, \alpha\}$: HMC Parameters

Output: N_s Posterior Samples of Model Parameters

BEGIN ALGORITHM:

```

 $\theta_0 = \text{InitializeNetworkParameters}()$ 
 $\sigma_0^2 = \text{InitializeVarianceParameters}()$ 
 $m_{\text{prev}} \sim N(0,1)$ 
FOR i in 1 to  $N_s$  DO:
     $\sigma_i^2 = \text{GibbsUpdate}(\theta_{i-1})$ 
     $\gamma_0 = \theta_{i-1}$ 
     $m_0 = \text{InitializeMomentum}(\alpha, m_{\text{prev}})$ 
    FOR t in 1 to L DO:
         $m_* = m_{t-1} + \frac{\epsilon}{2} \nabla_{\gamma} P(\gamma_{t-1} | X, Y; \sigma_i^2)$ 
         $\gamma_t = \gamma_{t-1} + \epsilon * m_*$ 
         $m_t = m_* + \frac{\epsilon}{2} \nabla_{\gamma} P(\gamma_t | X, Y; \sigma_i^2)$ 
    END
     $m_{\text{prev}} = m_t$ 
    IF  $\text{Accept}(\gamma_t)$ :  $\theta_i = \gamma_t$ 
    ELSE:  $\theta_i = \theta_{i-1}$ 
END

```

Figure 1: Algorithm for HMC-based posterior sampling for the neural network model.

A few details in the HMC algorithm as shown need further explanation. First the momentum variables (m) are refreshed after every sequence of L leap-frog updates, shown in the algorithm

as `InitalizeMomentum(α)`. In the most simple formulation each momentum component is a independent draw from a Normal distribution, with mean 0 and standard deviation of 1. However, this can lead to wasted computation because the sampler may start out in bad direction by chance, requiring more leap-frog updates until the sampler is heading in a useful direction resulting in random-walk like behavior. To combat this, we use the persistent momentum refreshes [Nabney, 2002, Neal, 1995] which initializes the momentum using a weighted combination of the final momentum value of the previous leap-frog update and a draw of standard normal random variable. Using the notation of the algorithm, this is shown below:

$$m_0 = \alpha * m_{prev} + \sqrt{1 - \alpha^2} * \zeta$$

where $\zeta \sim N(0, 1)$. If the proposal is rejected the momentum must be negated. This must be done to ensure that the canonical distribution is left intact [Nabney, 2002]. This formulation reduces the number of leap-frog updates (L) needed to reach a distant point by suppressing random-walk behavior while leaving the correct target distribution of the Markov chain intact. [Nabney, 2002, Neal, 1995, Neal, 2011a]. The `Accept(γ_t)` function returns true if the new proposal, γ_t , is accepted according to a modified Metropolis-Hastings acceptance probability. `Accept(γ_t)` returns true with the following probability:

$$\bar{\alpha} = \min \left(1, \frac{P(\gamma_t | X, Y; \sigma_i^2) - \frac{1}{2} m_t^T m_t}{P(\gamma_{t-1} | X, Y; \sigma_i^2) - \frac{1}{2} m_{t-1}^T m_{t-1}} \right)$$

However, the posterior distribution is often very ‘bumpy’ with many posterior modes [Neal, 1995]. This property may be exacerbated in high dimensions, so becoming stuck in one mode for extended periods of time is a great concern. To alleviate this we modify the acceptance probability, $\bar{\alpha}$, to correspond to a *flattened* version of the posterior whose acceptance probability is given as $\alpha^* = \bar{\alpha} \cdot T$, for $T > 1$, which is equivalent to sampling from $P(\theta | X, Y; \sigma_i^2)^{\frac{1}{T}}$. While its true that we are no longer sampling from the exact posterior $P(\theta | X, Y; \sigma_i^2)$, under mild regularity conditions the posterior modes of the correct distribution remain intact [Andrieu et al., 2003]. Since none of the parameters have biological interpretations modifying the posterior in this way of little concern, if the full procedure is capable of maintaining a favorable discriminatory ability. We find the trade-off between ease of sampling across a wide-range possible scenarios and exactness of the posterior to be acceptable.

2.2 HMC Using Graphics Processing Units (GPUs)

Previous work has shown how the gradient and log-posterior evaluations needed by HMC can be sped-up by as much as 150x for large problems using Graphics Processing Units (GPUs) [Beam et al., 2014]. We adopt that framework here and express the gradient calculations as matrix-vector operations or element-wise operations. Similarly, evaluation of the log-posterior can be expressed in terms of linear operators and element-wise operations. Using GPUs for these operations is well known in the neural network literature [Bergstra et al., 2010, Lopes and Ribeiro, 2009, Oh and Jung, 2004] as the gradient of the log-posterior corresponds roughly to the well-known back-propagation algorithm and evaluation of the log-posterior corresponds to the feed-forward operation in standard neural networks. However, to our knowledge this study represents the first GPU-enabled implementation of Bayesian neural networks. Without GPU computing, it is likely that the computational burden imposed by large datasets would be too great for the Bayesian neural network framework to be feasible.

All of methods discussed in this study are implemented in the Python programming language. All GPU operations were conducted using the Nvidia CUDA-GPU [Nvidia, 2008] programming environment and accessed from Python using the PyCuda library [Klockner et al., 2012]. Source code containing the Bayesian neural network package is available at <https://github.com/beamandrew/BNN>.

2.3 Bayesian Test of Significance for ARD Parameters

Given the ARD prior a natural question to ask is how large do values of σ_j^2 need to be for input j to be considered relevant compared to a variable that is completely unrelated. This question can be framed in terms of a Bayesian hypothesis test. In this framework we will assume that under

the null hypothesis a variable is completely irrelevant in determining the status of the response. If this were a simple linear model, this would be equivalent to saying the regression coefficient for this variable has a posterior mean of zero. In the neural network model the ARD parameters that determine how relevant each input is are strictly positive, so we need a baseline or null model for the ARD parameters in order to determine if we can reject this null hypothesis of irrelevance. In order to construct and test this hypothesis, we make a simplifying assumption that weights for unrelated variables have a normal distribution with mean 0 and variance σ_{null}^2 i.e. $w_{kj} \sim N(0, \sigma_{null}^2)$. Due to the complex statistical model, the true posterior distributions for the weights under the null may not be exactly normal, but this approximation will be useful in simplifying the calculations. Additionally since, the prior for each weight is normal, this approximation will most likely not be too far from true posterior form under the stated null hypothesis.

Since σ_{null}^2 represents the null ARD parameter associated with a variable of no effect, we wish to test whether a variable of interest is significantly greater than this null value. We use the phrases null and significance here because of their familiar statistical connotations, but they should not be confused with the p-value based frequentist hypothesis testing procedure, as we are operating within a fully Bayesian framework. Our goal becomes testing whether the mean, μ_j of the posterior distribution for the ARD parameter, σ_j^2 , is greater than the mean of the null, μ_{null} , for the null ARD parameter σ_{null}^2 . Specifically we wish to test the following null hypothesis:

$$H_0 : \mu_{null} = \mu_j$$

against the one-sided alternative:

$$H_a : \mu_{null} < \mu_j$$

To test this, we need to know the closed form of μ_{null} . Making use of the iterative two-stage sampling scheme, we will derive this form by induction. We will also make use of several well-known facts of random variables. Firstly, if a random variable X has an inverse-gamma distribution, i.e. $X \sim IG(\alpha, \beta)$, then the mean or expected value of X , $E[X]$, is given by $\frac{\beta}{\alpha-1}$. Next, if the sequence of random variables $X_1 \dots X_n$ are each independently and identically distributed as $N(0, \sigma^2)$, then $\sum_{i=1}^n \left(\frac{X_i}{\sigma}\right)^2 = < \frac{X_1}{\sigma}, \dots, \frac{X_n}{\sigma} >^T < \frac{X_1}{\sigma}, \dots, \frac{X_n}{\sigma} > \sim \chi_n^2$, i.e. a chi-squared random variable with n degrees of freedom. This sum has an expected value of n , from the definition of a chi-squared random variable. This implies the conditional expected value $E[< X_1, \dots, X_n >^T < X_1, \dots, X_n > | \sigma] = \sigma^2 * n$. Using these basic facts we will show that under the null, the two-stage sampling scheme leaves expected value of the ARD parameter invariant, i.e. $\mu_{null} = \mu_{prior}$.

For a network with h hidden units, let $w_j = < w_{1j}, \dots, w_{hj} >$ be a vector containing all of the weights associated with input j , where each component of w_j is initially distributed according to the prior, $N(0, \sigma_j^2)$ and $\sigma_j^2 \sim IG(\alpha_0, \beta_0)$. The mean, for σ_j at the start of the simulation is $\mu_0 = \frac{\beta_0}{\alpha_0-1}$. We begin the simulation at iteration $i=1$ and perform a Gibbs update of σ_j^2 . The Gibbs update for the shape parameter, $\alpha_1 = \alpha_0 + n_w/2$, is iteration independent and will remain fixed for the entirety of the simulation. However, the Gibbs update for the scale parameter, $\beta_1 = \beta_0 + (w_j^T w_j)/2$, depends upon the current values of the weights, and thus will take on a random value at each iteration. However, we can compute the expected value for β_1 as:

$$\begin{aligned} E[\beta_1] &= E \left[\beta_0 + \frac{w_j^T w_j}{2} \right] \\ &= \beta_0 + \frac{1}{2} E[w_j^T w_j] \\ &= \beta_0 + \frac{1}{2} (h \cdot E[\sigma_0^2]) \\ &= \beta_0 + \frac{h}{2} \cdot \frac{\beta_0}{\alpha_0 - 1} \\ &= \beta_0 + \frac{h}{2} \cdot \mu_0 \end{aligned}$$

Thus, the expected value of β_1 after the first Gibbs update is $\beta_0 + \frac{h}{2} \cdot \mu_0$. Note that this expectation is independent of simulation iteration, so this result will hold for all $\beta_1, \beta_2, \dots, \beta$. Next, we use this fact to compute the expected value of the ARD parameter, $E[\sigma_1^2]$:

$$\begin{aligned}
E[\sigma_1^2] &= E \left[\frac{\beta_1}{\alpha_0 + h/2 - 1} \right] \\
&= \frac{E[\beta_1]}{\alpha_0 + h/2 - 1} \\
&= \frac{\beta_0 + h/2 \cdot \mu_0}{\alpha_0 + h/2 - 1} \\
&= \frac{\beta_0}{\alpha_0 - 1 + h/2} + \frac{h/2 \cdot \mu_0}{\alpha_0 - 1 + h/2} \\
&= \frac{\frac{1}{\alpha_0 - 1}}{\frac{1}{\alpha_0 - 1}} \cdot \frac{\beta_0}{\alpha_0 - 1 + h/2} + \frac{h/2 \cdot \mu_0}{\alpha_0 - 1 + h/2} \\
&= \frac{\mu_0}{1 + h/2 \cdot \frac{1}{\alpha_0 - 1}} + \frac{h/2 \cdot \mu_0}{\alpha_0 - 1 + h/2} \\
&= \mu_0 \left(\frac{1}{1 + h/2 \cdot \frac{1}{\alpha_0 - 1}} + \frac{h/2}{\alpha_0 - 1 + h/2} \right) \\
&= \mu_0 \left(\frac{\alpha_0 - 1}{\alpha_0 - 1 + h/2} + \frac{h/2}{\alpha_0 - 1 + h/2} \right) \\
&= \mu_0 \left(\frac{\alpha_0 - 1 + h/2}{\alpha_0 - 1 + h/2} \right) \\
&= \mu_0
\end{aligned}$$

Thus, the Gibbs update of the ARD parameter does not change the expected value under the null, since we defined $E[\sigma_0^2] = \mu_0$. This establishes the base case, and now we show the induction step. Given $E[\beta_{t+1}] = E[\beta_t] = \beta_0 + \frac{h}{2} \cdot \mu_0$ and $E[\sigma_t] = \mu_0$ then:

$$\begin{aligned}
E[\sigma_{t+1}] &= E \left[\frac{\beta_{t+1}}{\alpha_0 + h/2 - 1} \right] \\
&= \frac{E[\beta_{t+1}]}{\alpha_0 + h/2 - 1} \\
&= \frac{\beta_0 + h/2 \cdot \mu_0}{\alpha_0 + h/2 - 1} \\
&= \mu_0
\end{aligned}$$

where the simplification between lines 3 and 4 proceeds as before. This concludes the proof.

3 Results

3.1 Existing Methods Used for Comparison

We selected several methods to serve as baselines for evaluation of the BNNs performance. As previously mentioned BEAM and MDR are widely used methods and so were included in our evaluation. We used a custom compiled 64-bit version of BEAM using the source provided on the website [Zhang, 2014] of the authors of [Zhang and Liu, 2007]. The java-based MDR package was downloaded from the MDR source-forge repository (<http://sourceforge.net/projects/mdr/>) and called from within a Python script. To evaluate the effectiveness of tree-based methods, we used an approach nearly identical to that in [Li et al., 2011], which was based on boosted decision trees. The boosted decision tree model provides measures of relative influence for each variable that indicate

how important a given variable is, relative to the others in the model. To fit the boosted tree model we used the `gbm` package in R. Finally, we also included the standard 2 degrees-of-freedom chi-square test of marginal effects.

As discussed, some approaches such as MDR and GBM require a permutation testing strategy to assess statistical significance. This makes assessing their performance on large datasets difficult, due to the amount time required to perform the permutation test. During our pilot investigations on a dataset containing 1,000 SNPs, each individual run of MDR was found to take roughly 1 minute to complete. The time needed to complete the required 20,000 permutations would be roughly 2 weeks. If we wish to evaluate a methods effectiveness on hundreds or thousands of such datasets, this run time becomes prohibitive. As such, we divided our primary analysis into two sections. In the first section, we evaluated methods that *do not* rely on permutation testing on datasets containing 1,000 SNPs each. However, since we wish to compare the results of the BNN to that of MDR and GBM, we performed a second set of analyses on smaller datasets that only contained 50 SNPs each, for which permutation testing is feasible. This two-pronged strategy allowed us to evaluate a wide range of popular approaches in a reasonable amount of time, while serving to underscore the need for methods that do not rely on permutation testing.

3.2 Parametric Models of Multi-Locus Relationships

In this section we performed an analysis of three biallelic models of genotypic relationships. These models have been used previously [Zhang and Liu, 2007, Li et al., 2011] and are meant to reflect theoretical and empirical evidence for genetic relationships involving multiple loci [Li and Reich, 2000]. Tables 1, 2, and 3 contain the risk relative of disease for each genotype combination, where a capital and lower case letters represent the major and minor alleles, respectively.

Table 1: Additive Risk Model

Genotype	AA	Aa	aa
BB	η	$\eta(1 + \theta)$	$\eta(1 + 2\theta)$
Bb	$\eta(1 + \theta)$	$\eta(1 + 2\theta)$	$\eta(1 + 3\theta)$
bb	$\eta(1 + 2\theta)$	$\eta(1 + 3\theta)$	$\eta(1 + 4\theta)$

Table 2: Threshold Risk Model

Genotype	AA	Aa	aa
BB	η	η	η
Bb	η	$\eta(1 + \theta)$	$\eta(1 + \theta)$
bb	η	$\eta(1 + \theta)$	$\eta(1 + \theta)$

Table 3: Epistatic Risk Model

Genotype	AA	Aa	aa
BB	η	η	$\eta(1 + 4\theta)$
Bb	η	$\eta(1 + 2\theta)$	η
bb	$\eta(1 + 4\theta)$	η	η

The symbols η and θ in the tables represent the baseline risk and effect size, respectively. We simulated genotypes for the disease SNPs for a range of minor allele frequencies (MAFs) and simulated the disease status for 1,000 cases and 1,000 controls using the risks given in Tables 1, 2, and 3. We embedded the causal SNPs in a background of 998 non-causal SNPs, for a total of 1,000 SNPs to be considered. For each combination of effect size, $\theta \in \{0.5, 1.0, 1.5, 2.0\}$,

MAF $\in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, and model type (Additive, Threshold and Epistasis) we generated 100 datasets. This yielded a total of 6,000 datasets for evaluation. All datasets in this section were created using the R statistical programming language [Team et al., 2005].

We ran BNN, BEAM, and the χ^2 test on each dataset and recorded whether or not both disease SNPs were declared as significant by each method. We took the fraction of datasets where both disease SNPs were correctly identified as an estimate of statistical power. For BEAM and the χ^2 test, we used the canonical Bonferroni corrected significance threshold of $p < 0.05$. We used the recommended parameter settings for BEAM [Zhang and Liu, 2007] and performed 1×10^6 sampling iterations for each dataset. For the BNN approach, we used a network with 1 hidden layer and 5 logistic units and a softmax output layer with 2 units. The network parameters in the hidden layer are given ARD priors, while the network parameters in the output are given a common Gaussian prior. The hyper parameters for the Inverse-Gamma prior for the ARD parameters were $\alpha_0 = 5, \beta_0 = 2$ while the hyper parameters for the Gaussian priors were $\alpha_0 = 0.1, \beta_0 = 0.1$. The parameters for the HMC algorithm were $\epsilon = 5 \times 10^{-2}$, $L = 15$, $\alpha = 0.75$, and $T = 5 \times 10^3$. The cutoff value for the novel Bayesian ARD testing framework was 0.6. We discarded the first 25 samples as burn-in and kept 100 samples to be used for inference. Processing of each dataset by the BNN took approximately 3 minutes. The results are shown below in Figures 2, 3 and 4.

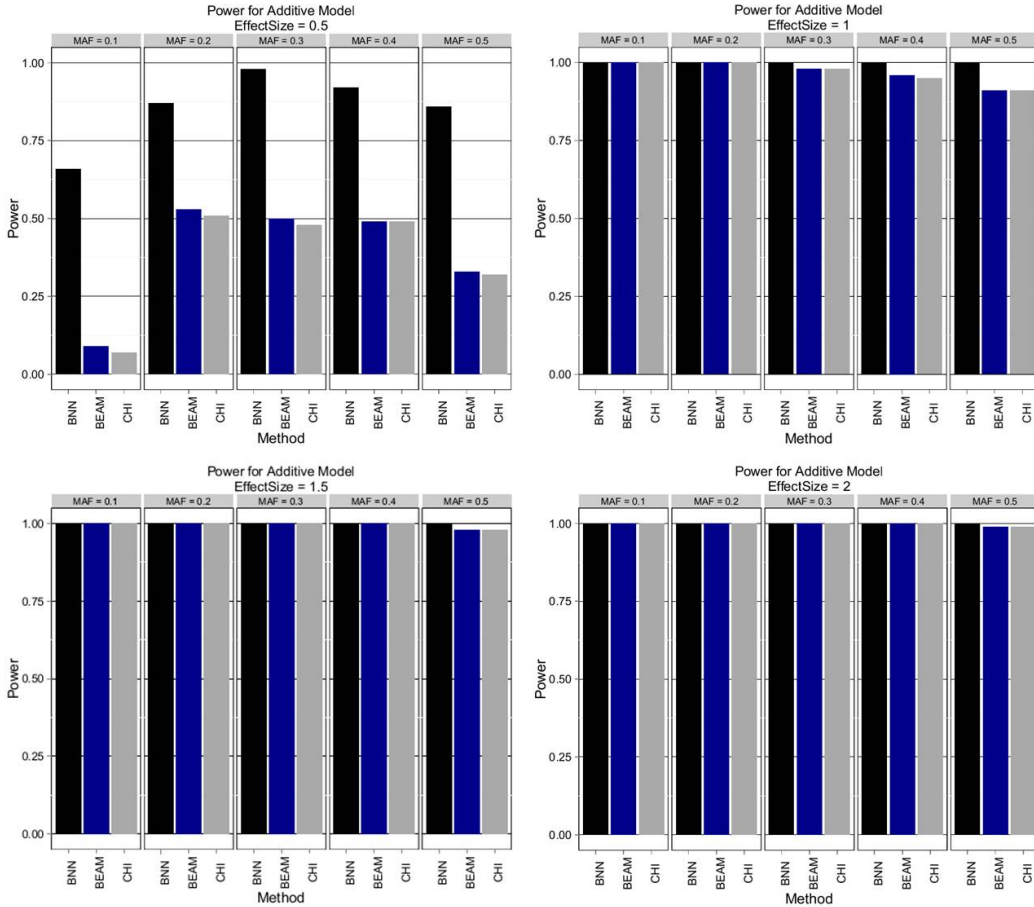


Figure 2: Additive Model. Estimated power to detect both disease SNPs using Bayesian neural networks (BNN), BEAM, and χ^2 test (CHI) with 2 d.f. Effect sizes of $\{0.5, 1.0, 1.5, 2.0\}$ are shown in order from left to right, top to bottom. Within each pane results are stratified by minor allele frequency (MAF).

BNNs were found to be uniformly more powerful than both BEAM and the χ^2 test for the additive model. BNNs show excellent power, even for small effect sizes and achieve 100% power for second smallest effect size across all tested MAFs. In contrast, BEAM showed relatively little power for the

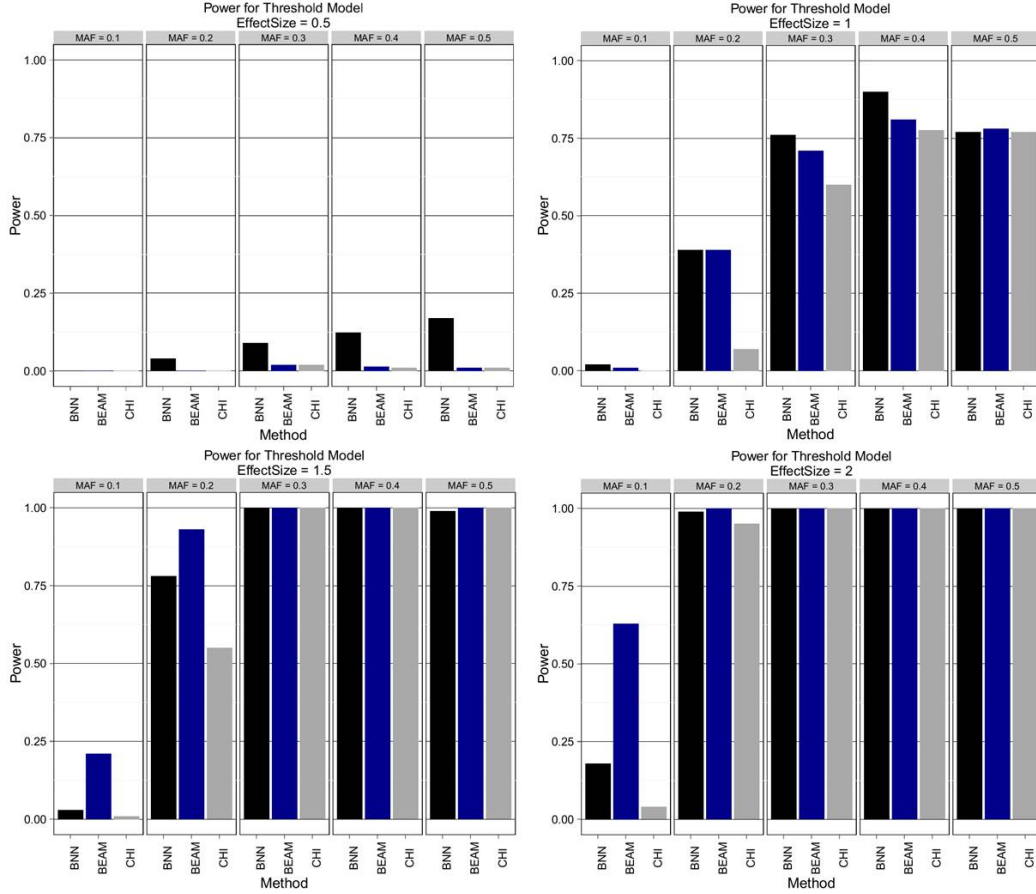


Figure 3: Threshold Model. Estimated power to detect both disease SNPs using Bayesian neural networks (BNN), BEAM, and χ^2 test (CHI) with 2 d.f. Effect sizes of {0.5, 1.0, 1.5, 2.0} are shown in order from left to right, top to bottom. Within each pane results are stratified by minor allele frequency (MAF).

smallest effect size and never achieves 100% for all MAFs, even at the highest level of effect size. The threshold model tells a similar story. For all but 3 combinations of MAF and effect size, the BNN model is again uniformly more powerful than both BEAM and the 2 test. The picture from the epistatic model is slightly more mixed. BEAM appeared to do a better job at the smallest effect size, while performing equally well as BNNs on the remaining three effect size levels. All three methods had almost no power to detect the causal SNPs for a MAF of 0.5. These results suggest that BNN is uniformly more powerful the χ^2 test for these genetic models, and may be more powerful than BEAM in most instances.

3.3 Simulated Epistatic Relationships without Marginal Effects

In this section, we evaluated the performance of all the methods examined in the previous section (BNN, BEAM, and the χ^2) as well as GBM and MDR. Since MDR and GBM rely on permutation testing, we reduced the size of the dataset to accommodate this strategy. To generate test datasets, we used the GAMETES software package [Urbanowicz et al., 2012]. This package allows users to specify the proportion of variance for case/control status that is due to genetic variants (i.e. broad-sense heritability) as well as how many loci are involved in determining trait status. These relationships are generated such that there are minimal marginal effects, resulting in relationships that are nearly purely epistatic. Relationships without marginal effects are in some sense ‘harder’ than those with marginal effects, because the causal SNPs contribute to trait status only through their interaction. Preliminary analysis on the reduced SNP datasets indicated that if the same models were

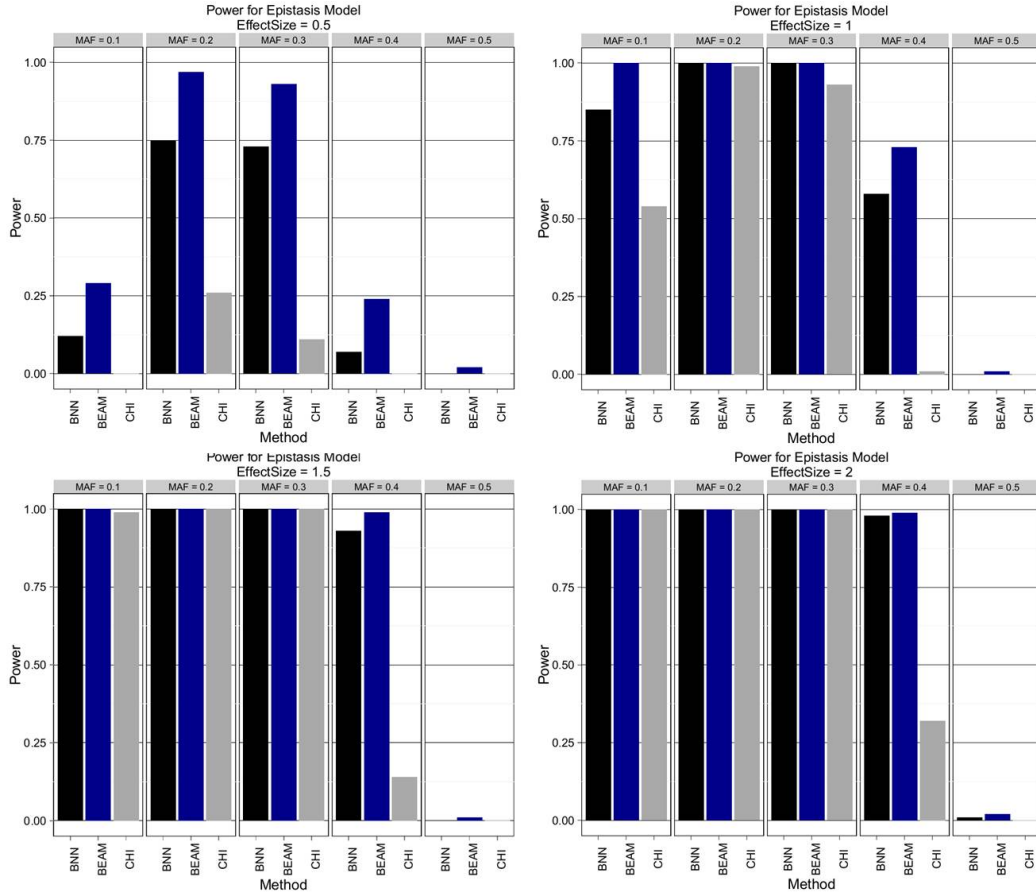


Figure 4: Epistatic Model. Estimated power to detect both disease SNPs using Bayesian neural networks (BNN), BEAM, and χ^2 test (CHI) with 2 d.f. Effect sizes of $\{0.5, 1.0, 1.5, 2.0\}$ are shown in order from left to right, top to bottom. Within each pane results are stratified by minor allele frequency (MAF).

used as in the previous section, most methods would have nearly 100% power for all simulated scenarios, which would provide little useful feedback for discerning which approaches were working best. This was the primary motivation for using the ‘harder’, purely epistatic relationships instead of the parametric models we used previously.

Using GAMETES, we analyzed two levels of heritability (5% and 10%) across a range of MAFs (0.05, 0.1, 0.2, 0.3, 0.4, 0.5). Power was measured as in the previous section using 100 instances for each heritability/MAF combination for a total of 1200 data sets used in evaluation. The results are shown below in Figures 5 and 6.

BNN outperform all methods from the previous section (BEAM and χ^2 test) by a very wide margin. This suggests that BEAM may be less robust to detect causal SNPs in the absence of marginal effects than previously thought, as it never achieves 25% power in any of the scenarios tested. Again, we find these results encouraging as they indicate that BNNs are indeed powerful relative to existing approaches. Additionally, BNN outperformed the GBM method in all but 2 scenarios, indicating that BNN may be more adept at detecting purely epistatic signals across a broad array of MAFs and effect sizes. MDR performs well across every parameter combination tested, but as mentioned previously it is incapable of performing this analysis on a GWAS scale due to the exhaustive search technique and the need to perform permutation testing to assess statistical significance. To conclude this section, we note that BNN was the only method that did well across a variety of genetic models, number of SNPs, and MAFs while being capable of scaling to GWAS-sized data. This provides

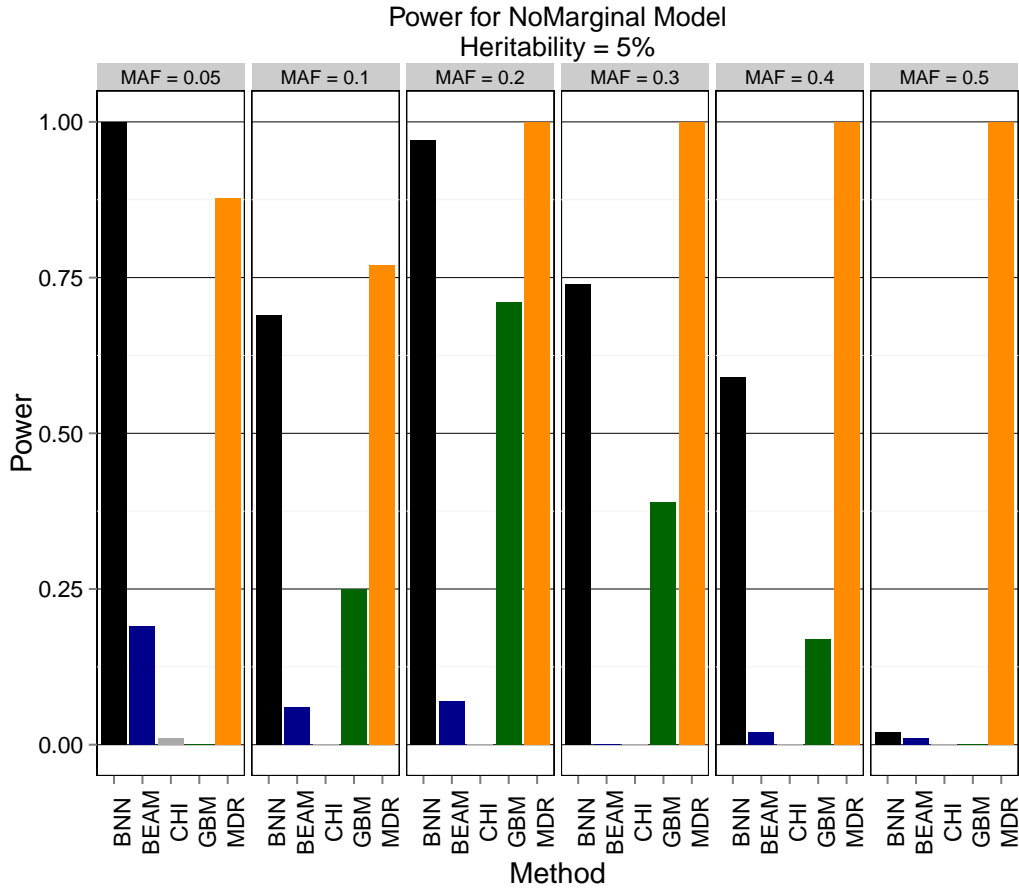


Figure 5: Purely Epistatic Model with 5% heritability. Estimated power to detect both disease SNPs of Bayesian neural networks (BNN), BEAM, 2 test (CHI) with 2 d.f., gradient boosted trees (GBM), and MDR. The results are stratified by minor allele frequency (MAF).

evidence that BNN framework is deserving of further investigation as an analysis technique for association studies.

3.4 Sensitivity and Specificity Analysis of the ARD Test

The cutoff value used for the ARD test has an obvious impact on the methods performance. In the extreme case, a cutoff of 0 would result in nothing being significant while a cutoff value of 1 would result in everything being declared as such. The cutoff value controls the tradeoff between sensitivity (i.e. the true positive rate) and specificity (i.e. the true negative rate, which is equivalent to 1 the false positive rate). Evaluation of the false positive rate for the cutoff value of 0.6 used in the previous experiments indicates that the BNN method properly controls the amount of false positives. We observed an average false positive rate (FPR) of roughly 0.005 and 0.06 for the parametric models and the purely epistatic models, respectively as shown in Figure 7.

To examine the trade off between the true positive rate (TPR) and FPR as the cutoff value is changed, we modulated the cutoff from 0 to 1 in increments of 0.01 and recorded the true positive and false positive rate for each data set in the two previous sections. In Figure 8, we averaged the TPR and FPR over effect size and MAF to produce a receiver-operator characteristic (ROC) curve for each of the 4 genetic models. The legend displays the area under the curve (AUC) for each model.

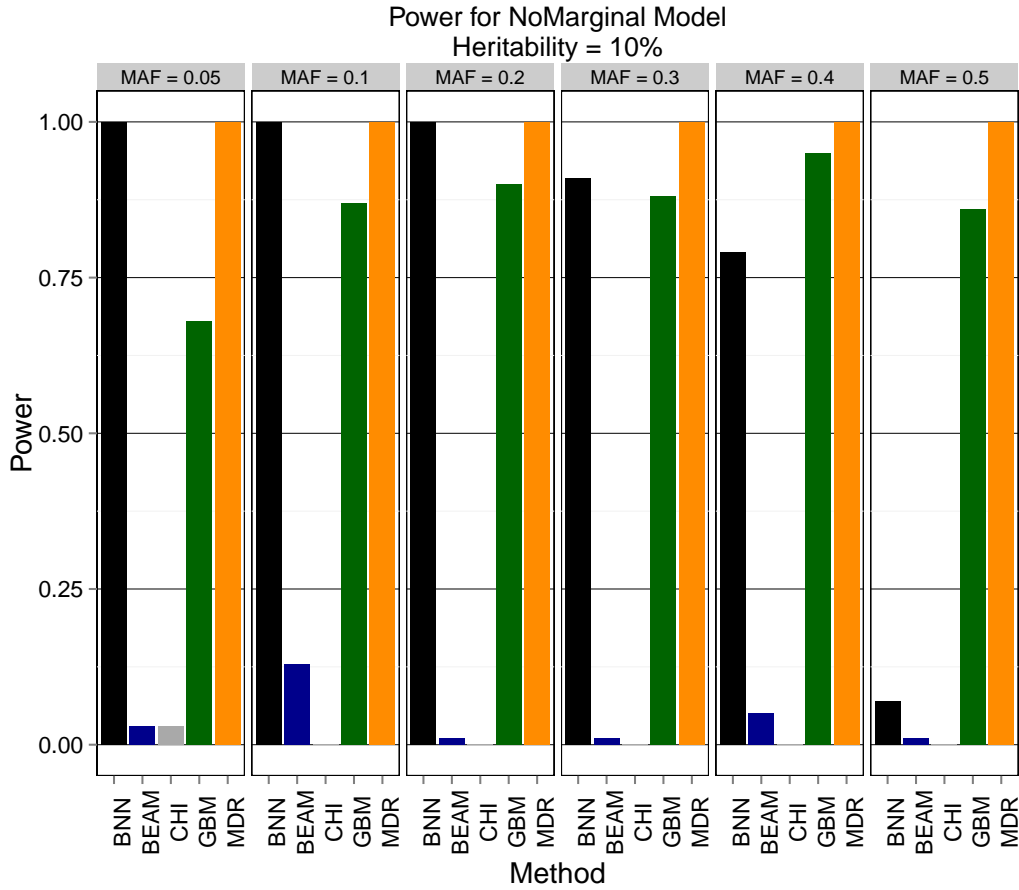


Figure 6: Purely Epistatic Model with 10% heritability. Estimated power to detect both disease SNPs of Bayesian neural networks (BNN), BEAM, 2 test (CHI) with 2 d.f., gradient boosted trees (GBM), and MDR. The results are stratified by minor allele frequency (MAF).

These results show that BNN-ARD test for variable importance is able to achieve a high true positive rate, while maintaining a low false positive rate, which is an indication the method is performing as well and as expected.

3.5 Analysis of Tuberculosis Data

To evaluate the performance of Bayesian neural networks on a real dataset, we analyzed a GWAS designed to find genetic markers associated with tuberculosis (TB) disease progression. The dataset describe in [Oki et al., 2011], contains information on roughly 60,000 SNPs and 105 subjects. For our study, each subject was classified as currently infected with any form of tuberculosis (i.e. extrapulmonary or pulmonary) or having a latent form of TB confirmed through a positive tuberculin skin test (purified protein derivative positive). Quality control was performed and SNPs with missing values were excluded, as were SNPs that were found to be out of Hardy-Weinberg equilibrium at the 0.05 level. After QC, there were 16,925 SNPs available for analysis and 104 subjects. Based on evidence of subpopulations in this data [Oki et al., 2011], subjects were assigned to one of three clusters created using the top two principal components and cluster membership was included as a covariate in the model. Sampling of the Bayesian neural network was conducted as outlined in the previous section, with ARD hyper-parameters of $\alpha_0 = 3$, $\beta_0 = 1$. We performed 100 burn-in iterations followed by 1,000 sampling iterations which took approximately 20 hours. The top five SNPs based on posterior ARD probabilities are shown below in Table 4. The SNP reported as the 2nd most significant in [Oki et al., 2011] (rs10490266) appeared in our analysis as the 31st most

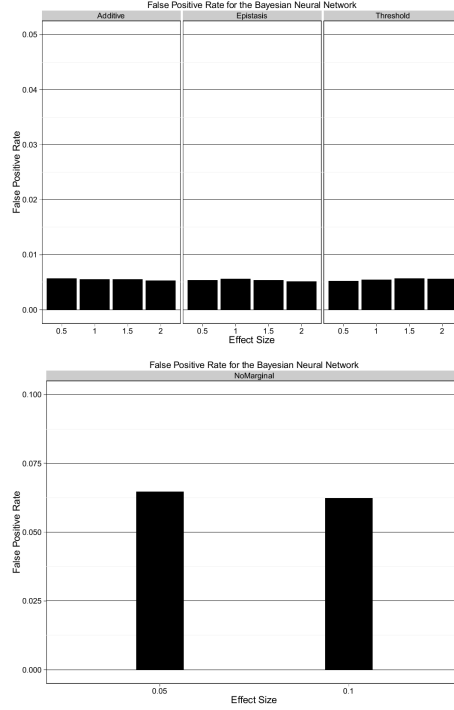


Figure 7: False Positive Rates (FPR) for each model/effect size combination, averaged over MAF.

Table 4: Top 5 SNPs based on posterior ARD probabilities. Note these probabilities are presented in terms of involvement (larger indicates a SNP is more likely to be involved).

SNP	CHR	$Pr(\mu_j > \mu_{null})$
rs966414	2	0.524
rs1378124	8	0.515
rs9327930	5	0.509
rs4721214	7	0.502
rs4721214	9	0.498

significant SNP. Only one of the SNPs in Table 4 is currently known to be located within a gene (rs1378124 - MATN2) according to dbSNP. Every SNP reported in Table 4 is located on a the same chromosome and within 10-50 MB of loci previously reported as having a statistically significant association with pulmonary tuberculosis susceptibility [Png et al., 2012] in an Indonesian population. The loci reported in [Png et al., 2012] were unfortunately either not part of the original SNP library or removed during the QC process in this study. Due to the small sample size of this dataset, it is hard to say conclusively which of the SNPs reported here and in [Oki et al., 2011] are most likely to replicate in a larger study. However, we present this analysis to demonstrate that the BNN framework is capable of analyzing data sets containing a high number of SNPs in a relatively short amount time.

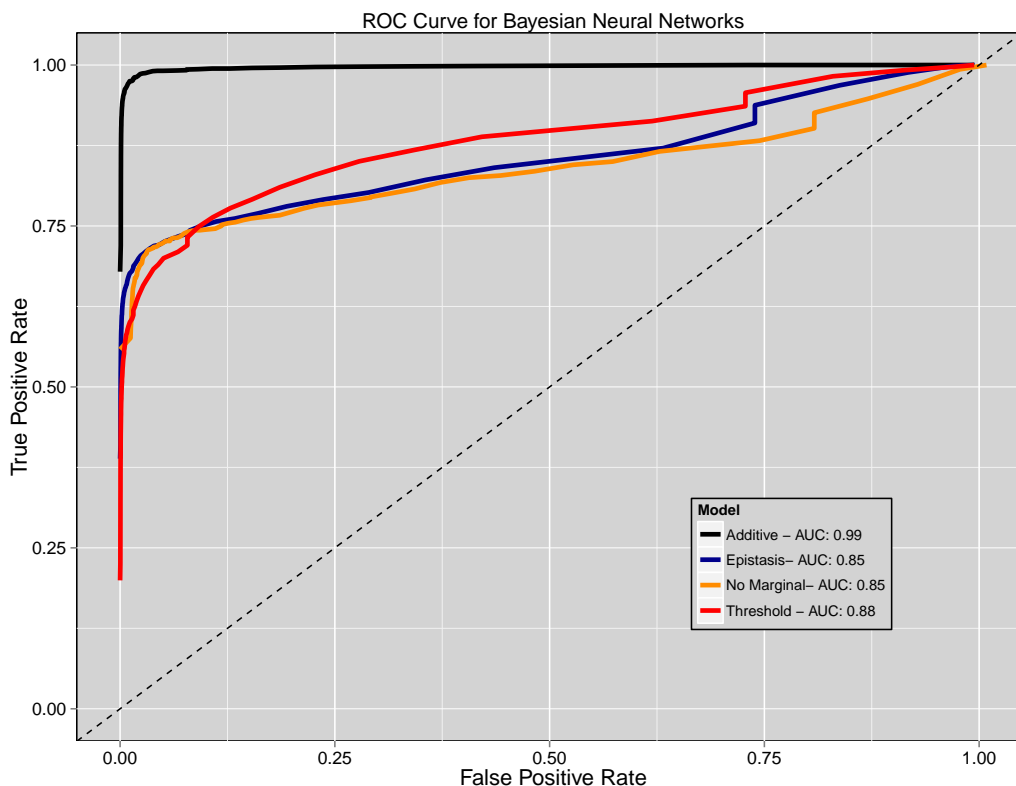


Figure 8: Receiver-Operator Characteristic (ROC) curve for BNNs. Each line represents the ROC curve for a different genetic model, averaged over effect size and MAF. The area under the curve (AUC) for each model is shown in the legend.

4 Conclusions

In this study we have proposed the use of Bayesian neural networks for association studies. This approach was shown to be powerful across a broad spectrum of different genetic architectures, effect sizes, and MAFs. Of the approaches that do not rely on permutation testing, BNN was uniformly more powerful than the standard χ^2 test and almost uniformly more powerful than the popular BEAM method in the scenarios considered. BNN again showed a near uniformly better performance than the GBM method. MDR was very competitive with BNN in our evaluations, however MDR is incapable of scaling to larger datasets due to both its exhaustive search technique and reliance on permutation testing. In conclusion, we have demonstrated that BNNs are a powerful technique for association studies while having the capability of scaling to large GWAS sized datasets.

Availability of Code

Source code implementing the GPU-based Bayesian neural network framework outlined in this paper is available at <https://github.com/beamandrew/BNN>.

References

- [Andrieu et al., 2003] Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43.
- [Baesens et al., 2002] Baesens, B., Viaene, S., den Poel, D. V., Vanthienen, J., and Dedene, G. (2002). Bayesian neural network learning for repeat purchase modelling in direct marketing. *European Journal of Operational Research*, 138(1):191–211.

- [Beam et al., 2014] Beam, A. L., Ghosh, S. K., and Doyle, J. (2014). Fast hamiltonian monte carlo using gpu computing. *ArXiv e-prints*. 1402.4089; Provided by the SAO/NASA Astrophysics Data System.
- [Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.
- [Bergstra et al., 2010] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. In *Proceedings of the Python for Scientific Computing Conference, SciPy*.
- [Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.
- [Diaz-Uriarte and de Andres, 2006] Diaz-Uriarte, R. and de Andres, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7:3. LR: 20130905; JID: 100965194; OID: NLM: PMC1363357; 2005/07/08 [received]; 2006/01/06 [accepted]; 2006/01/06 [aheadofprint]; epublish.
- [Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine.(english summary). *Ann.Statist*, 29(5):1189–1232.
- [Greene et al., 2010] Greene, C. S., Sinnott-Armstrong, N. A., Himmelstein, D. S., Park, P. J., Moore, J. H., and Harris, B. T. (2010). Multifactor dimensionality reduction for graphics processing units enables genome-wide testing of epistasis in sporadic als. *Bioinformatics*, 26(5):694–695.
- [Guyon et al., 2002] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.
- [Hahn et al., 2003] Hahn, L. W., Ritchie, M. D., and Moore, J. H. (2003). Multifactor dimensionality reduction software for detecting gene–gene and gene–environment interactions. *Bioinformatics*, 19(3):376–382.
- [Hastings, 1970] Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.
- [Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- [Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.
- [Jiang et al., 2011] Jiang, X., Neapolitan, R. E., Barmada, M. M., and Visweswaran, S. (2011). Learning genetic epistasis using bayesian network scoring criteria. *BMC bioinformatics*, 12:89–2105–12–89. LR: 20131018; GR: 1K99LM010822-01/LM/NLM NIH HHS/United States; GR: K99 LM010822/LM/NLM NIH HHS/United States; GR: R00 LM010822/LM/NLM NIH HHS/United States; GR: R01 HG004718/HG/NHGRI NIH HHS/United States; JID: 100965194; OID: NLM: PMC3080825; 2010/07/21 [received]; 2011/03/31 [accepted]; 2011/03/31 [aheadofprint]; epublish.
- [Klockner et al., 2012] Klockner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., and Fasih, A. (2012). Pycuda and pyopencil: A scripting-based approach to gpu run-time code generation. *Parallel Computing*, 38(3):157–174.
- [Koo et al., 2013] Koo, C. L., Liew, M. J., Mohamad, M. S., and Salleh, A. H. M. (2013). A review for detecting gene-gene interactions using machine learning methods in genetic epidemiology. *BioMed research international*, 2013.
- [Li et al., 2011] Li, J., Horstman, B., and Chen, Y. (2011). Detecting epistatic effects in association studies at a genomic level based on an ensemble approach. *Bioinformatics*, 27(13):222–229.
- [Li and Reich, 2000] Li, W. and Reich, J. (2000). A complete enumeration and classification of two-locus disease models. *Human heredity*, 50(6):334–349.
- [Li et al., 2002] Li, Y., Campbell, C., and Tipping, M. (2002). Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, 18(10):1332–1339.
- [Lisboa et al., 2003] Lisboa, P. J., Wong, H., Harris, P., and Swindell, R. (2003). A bayesian neural network approach for modelling censored data with an application to prognosis after surgery for breast cancer. *Artificial Intelligence in Medicine*, 28(1):1–25.

- [Lopes and Ribeiro, 2009] Lopes, N. and Ribeiro, B. (2009). *GPU implementation of the multiple back-propagation algorithm*, pages 449–456. Intelligent Data Engineering and Automated Learning-IDEAL 2009. Springer.
- [Lunetta et al., 2004] Lunetta, K. L., Hayward, L. B., Segal, J., and Eerdewegh, P. V. (2004). Screening large-scale association study data: exploiting interactions using random forests. *BMC genetics*, 5(1):32.
- [Manolio et al., 2009] Manolio, T. A., Collins, F. S., Cox, N. J., Goldstein, D. B., Hindorff, L. A., Hunter, D. J., McCarthy, M. I., Ramos, E. M., Cardon, L. R., Chakravarti, A., et al. (2009). Finding the missing heritability of complex diseases. *Nature*, 461(7265):747–753.
- [Metropolis et al., 2004] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (2004). Equation of state calculations by fast computing machines. *The Journal of chemical physics*, 21(6):1087–1092.
- [Moore et al., 2006] Moore, J. H., Gilbert, J. C., Tsai, C.-T., Chiang, F.-T., Holden, T., Barney, N., and White, B. C. (2006). A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of hu disease susceptibility. *Journal of theoretical biology*, 241(2):252–261.
- [Motsinger-Reif et al., 2008a] Motsinger-Reif, A. A., Dudek, S. M., Hahn, L. W., and Ritchie, M. D. (2008a). Comparison of approaches for machine-learning optimization of neural networks for detecting gene-gene interactions in genetic epidemiology. *Genetic epidemiology*, 32(4):325–340.
- [Motsinger-Reif et al., 2008b] Motsinger-Reif, A. A., Reif, D. M., Fanelli, T. J., and Ritchie, M. D. (2008b). A comparison of analytical methods for genetic association studies. *Genetic epidemiology*, 32(8):767–778.
- [Motsinger-Reif and Ritchie, 2008] Motsinger-Reif, A. A. and Ritchie, M. D. (2008). Neural networks for genetic epidemiology: past, present, and future. *BioData mining*, 1(3).
- [Nabney, 2002] Nabney, I. (2002). *NETLAB: algorithms for pattern recognition*. Springer.
- [Neal, 2011a] Neal, R. (2011a). Mcmc for using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, pages 113–162.
- [Neal, 2011b] Neal, R. (2011b). Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, pages 113–162.
- [Neal, 1992] Neal, R. M. (1992). Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical report, Citeseer.
- [Neal, 1995] Neal, R. M. (1995). Bayesian learning for neural networks.
- [Neal, 1998] Neal, R. M. (1998). Assessing relevance determination methods using delve. *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, 168:97–132.
- [Nvidia, 2008] Nvidia, C. (2008). Programming guide.
- [Oh and Jung, 2004] Oh, K.-S. and Jung, K. (2004). Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314.
- [Oki et al., 2011] Oki, N. O., Motsinger-Reif, A. A., Antas, P. R., Levy, S., Holland, S. M., and Sterling, T. R. (2011). Novel human genetic variants associated with extrapulmonary tuberculosis: a pilot genome wide association study. *BMC research notes*, 4(1):28.
- [Png et al., 2012] Png, E., Alisjahbana, B., Sahiratmadja, E., Marzuki, S., Nelwan, R., Balabanova, Y., Nikolayevskyy, V., Drobniewski, F., Nejentsev, S., Adnan, I., et al. (2012). A genome wide association study of pulmonary tuberculosis susceptibility in indonesians. *BMC medical genetics*, 13(1):5.
- [Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA.
- [Team et al., 2005] Team, R. C. et al. (2005). R: A language and environment for statistical computing. *R foundation for Statistical Computing*.
- [Urbanowicz et al., 2012] Urbanowicz, R. J., Kiralis, J., Sinnott-Armstrong, N. A., Heberling, T., Fisher, J. M., and Moore, J. H. (2012). Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData mining*, 5(1):1–14.

- [Williams, 1995] Williams, P. M. (1995). Bayesian regularization and pruning using a laplace prior. *Neural computation*, 7(1):117–143.
- [Zhang, 2014] Zhang, Y. (2014). Academic website for yu zhang @online.
- [Zhang and Liu, 2007] Zhang, Y. and Liu, J. S. (2007). Bayesian inference of epistatic interactions in case-control studies. *Nature genetics*, 39(9):1167–1173.